University of Glasgow
DEPARTMENT OF
AEROSPACE
ENGINEERING

yroplane Derivative Identification Using a Matlab Routine

Vassilios M. Spathopoulos
Dr. Douglas Thomson
Dr. Stewart Houston

September 1997 Aero Dept Int. Rep No. 9715

Gyroplane Derivative Identification Using a Matlab Routine

Vassilios M. Spathopoulos
Dr. Douglas Thomson
Dr. Stewart Houston

September 1997 Aero Dept Int. Rep No. 9715

Department of Aerospace Engineering
University of Glasgow
Glasgow
G12 8QQ
U.K.

Abstract

In the following report it is described how a routine implemented in the **Matlab** software package is used for identifying the stability derivatives of a gyroplane. The software is designed to apply a frequency domain least squares approach in order to estimate the longitudinal and lateral force and moment derivatives of the aircraft. The aircraft system itself is represented as a linear state space model. The time histories of the system states in conjunction with the pilot input are analysed in order to perform the identification.

Initially the time series are transformed into the frequency domain. Linear regression is then performed over a suitable frequency range and an estimate of the stability derivatives together with the corresponding standard errors, is produced. The method is applied and tested against real data obtained from research conducted in the past within the department.

Nomenclature

$\underline{x}(t)$	state vector
$\underline{u}(t)$	input vector
A	state matrix
B	input matrix
V	covariance matrix
M_u, M_w etc.	moment derivatives
$\underline{y}$	output vector
$\underline{\theta}$	stability derivative vector
F_k	discrete frequency
k	integer
N	number of Fourier transform points
T	sampling interval
X,Y,Z	aircraft forces
L,M,N	aircraft moments

1. Introduction

The purpose of this report is to present details of a new software package for the identification of aircraft stability derivatives from flight data. In the identification analysis, the gyroplane system is represented by two sets of state space equations, defining the longitudinal and lateral dynamics of the aircraft. An equation-error method was applied in the frequency domain, using a least squares approach, in order to estimate the stability derivatives. The flight test data that was used was from the **VPM M16** gyroplane, and was obtained from a previous research project that was undertaken in the department[1]. The data acquisition system used for this application, functioned at a rate of 10Hz.The aim of the **Matlab** function that was designed was to automate all the necessary procedures by receiving the flight data as an input and outputting the stability derivatives and the associated standard errors (confidence levels) directly to the user. The way in which this was achieved will be presented in the following sections.

2. Theoretical Background

In this section, a brief description of the frequency domain, equation-error approach that was implemented in the **Matlab** function, will be presented. Assuming zero process noise, the model structure can be presented in the following way:

---

[1] The research was conducted under the CAA contract no. 7D/S/1125 by Dr. Stewart Houston and Dr. Douglas Thomson of Glasgow university aerospace department.

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t)$$

where,

A and B represent the stability and control derivatives respectively. Transforming into the frequency domain gives :

$$j\omega\underline{X}(\omega) = A\underline{X}(\omega) + B\underline{U}(\omega)$$

For example, if taking the pitching moment equation in the longitudinal dynamics subset, we obtain :

$$\dot{\boldsymbol{q}}(\boldsymbol{t}) = \boldsymbol{M}_{\boldsymbol{u}}\boldsymbol{u}(\boldsymbol{t}) + \boldsymbol{M}_{\boldsymbol{w}}\boldsymbol{w}(\boldsymbol{t}) + \boldsymbol{M}_{\boldsymbol{q}}\boldsymbol{q}(\boldsymbol{t}) + \boldsymbol{M}_{\theta}\theta(\boldsymbol{t}) + \boldsymbol{M}_{\Omega}\Omega(\boldsymbol{t}) + \boldsymbol{M}_{\eta_s}\eta_s(\boldsymbol{t})$$

By transforming into the frequency domain as for the general case :

$$\boldsymbol{j\omega Q}(\omega) = \boldsymbol{M}_{\boldsymbol{u}}\boldsymbol{U}(\omega) + \boldsymbol{M}_{\boldsymbol{w}}\boldsymbol{W}(\omega) + \boldsymbol{M}_{\boldsymbol{q}}\boldsymbol{Q}(\omega) + \boldsymbol{M}_{\theta}\Theta(\omega) + \boldsymbol{M}_{\Omega}\Omega(\omega) + \boldsymbol{M}_{\eta_s}\eta_s(\omega)$$

By separating the equation into real and imaginary parts, two sets of equations can be used to perform linear regression and to obtain the stability derivatives.

It is possible to implement ordinary regression to any equation of the form

$$\underline{y} = X\underline{\theta}$$

where $\underline{y}$ represents the estimates of the dependant variable, the matrix X involves values of the independent variables $\underline{x}_i$ arranged as columns and $\underline{\theta}$ represents the stability and control derivatives.[2]

The least squares solution for the vector $\underline{\theta}$ is

$$\underline{\hat{\theta}} = \left[X^T X\right]^{-1} X^T \underline{y}$$

The frequency range over which the data is used in the estimation process, must be selected on the basis of the intended application of the model. For most cases concerning the gyroplane application, a frequency range of 0.5 to 1 Hz was used. The results produced will be presented in Section 4 of the report. It should be noted at this point, that the data input to the **Matlab** function was initially processed through a kinematic consistency scheme. In the case of the examples used for this application, this was performed using a FORTRAN program already existing within the department. The way in which the above theory was applied in the software will be described in the following section.

3. Matlab Implementation

A listing of the **Matlab** coding used in the application, together with the appropriate

---

[2] For further reference, refer to 'A Frequency –Domain System Identification approach to Helicopter Flight Mechanics Model Validation', by C.G. Black and D.J. Murray-Smith, Vertica Vol. 13, 1989.

comments, can be found in **Appendix D**. It is initially assumed that the flight test data, including the system state time histories, is stored in a matrix *Amod*. The user must provide the function with this matrix together with the number of discrete Fourier transform points he wishes to use, N, and the frequency range f over which he wishes to analyse. These consist of the three inputs applied to the computer software.

The transformed frequency w is then calculated for the desired range by using the basic formula :

$$F_k=k/NT,$$

where N is the number of Fourier Transform points and T is the sampling interval.

The built in function *detrend.m* is then used in order to remove the mean value from each of the time histories. This is a necessary step to be performed before transforming into the frequency domain. The data is then transformed into the frequency domain by using the built in Fast Fourier Transform function *fft.m*. In order to perform the linear regression, the transformed pairs must be separated into their real and imaginary parts. This is performed by using the built in functions *real.m* and *imag.m*. Once the above are completed, the program is ready to perform the linear regression over each equation, and the user is asked to specify if the data to be used (i.e. the longitudinal or the lateral dynamics of the aircraft). The core part of the analysis is similar for both cases.

A matrix **A** of independent variables, matrix **B** of dependant variables and a covariance matrix **V** are all formed by using the data. In the case of the covariance matrix **V**, it is taken to be the identity matrix. Different values for **V** had little effect on the overall results. The core of the function is performed by the built in routine *lscov.m*. This outputs the least squares estimate for the stability derivatives and for their associated errors. Each force and moment equation is analysed in sequence and the

corresponding derivatives are identified and displayed on the screen. In the following section a series of examples will be presented to validate the routine.

4. Results

Once the routine was constructed, it was tested by using a series of data files for longitudinal and lateral manoeuvres. The same data had been used to identify the derivatives by using the Excel spreadsheet for previous research and the results were compared. One such example was produced by using the time series created by the frequency sweep of the longitudinal stick as shown in the diagram in **Figure 1**, in order to calculate the X, Z and M derivatives.

The screen output from the program can be found in **Appendix A**. It should be noted that the program outputs an estimate of the standard error associated with the estimation of each derivative, together with the estimation of the derivative itself. It also outputs a value for the coherence factor R^2 which is a measure of the credibility of the least squares fit.

The results were compared to those obtained using the Excel spreadsheet and proved to be almost identical.

The second case presented, involves the identification of the lateral Y and N derivatives which, as explained previously, can be derived by applying a rudder input to the system. The rudder input used is illustrated in **Figure 2**. The screen output from the program can be found in **Appendix B**.

The results were once more checked against the Excel method.

Finally, a data file created by a lateral stick input was used to identify the L derivatives. A diagram illustrating the input is presented in **Figure 3**. The screen output from the program can be found in **Appendix C**.

The results were once against consistent with the work done previously.

5. Conclusions

In this report, a detailed description was provided of a **Matlab** routine, designed to perform stability derivative identification using linear regression in the frequency domain. The description included presentation of the background theory applied, detailed coverage of the routine functionality and presentation of results. It is concluded that the software can be safely used as a staring point to any gyroplane identification application.

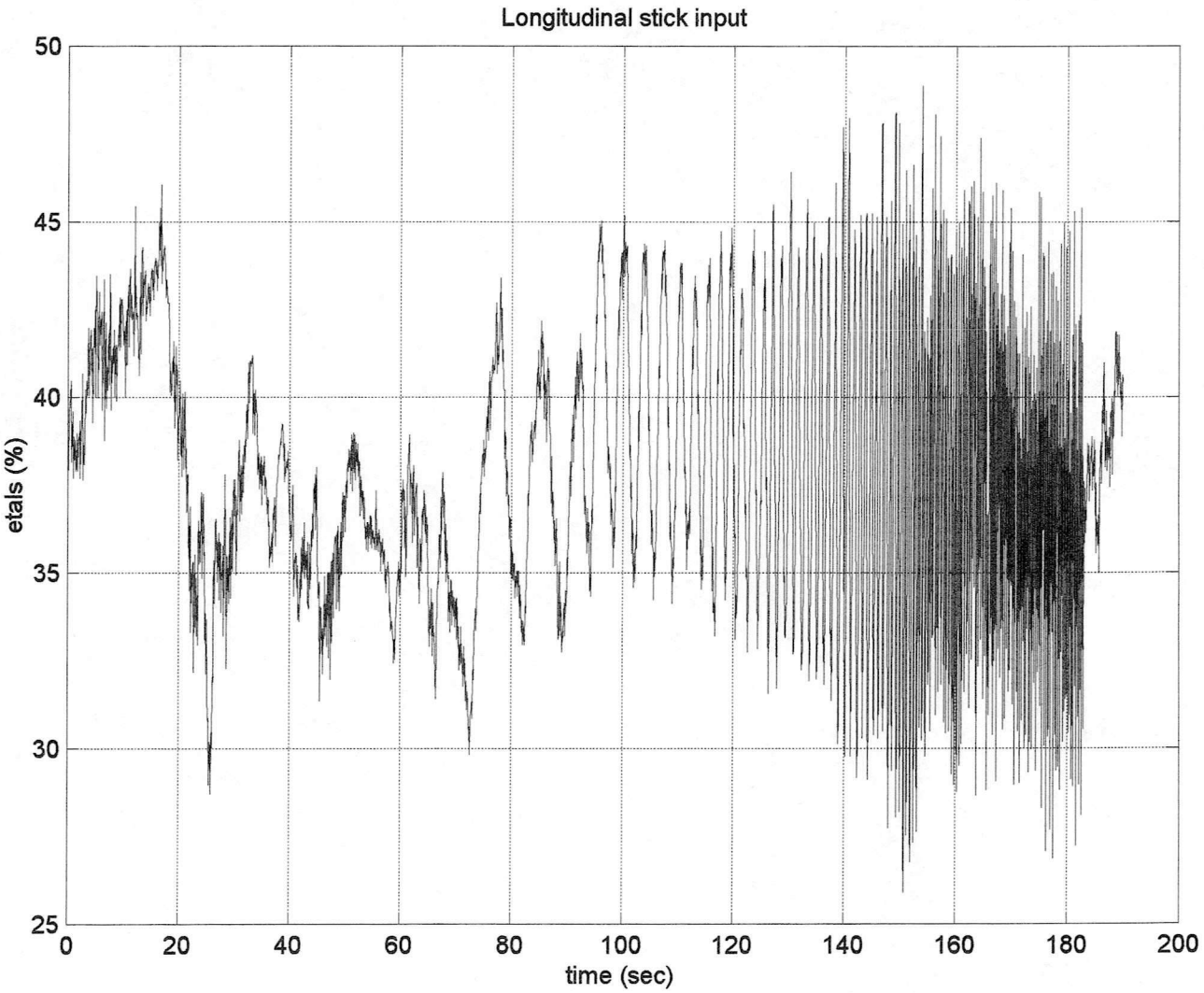

Figure 1 : Longitudinal stick input for case 1

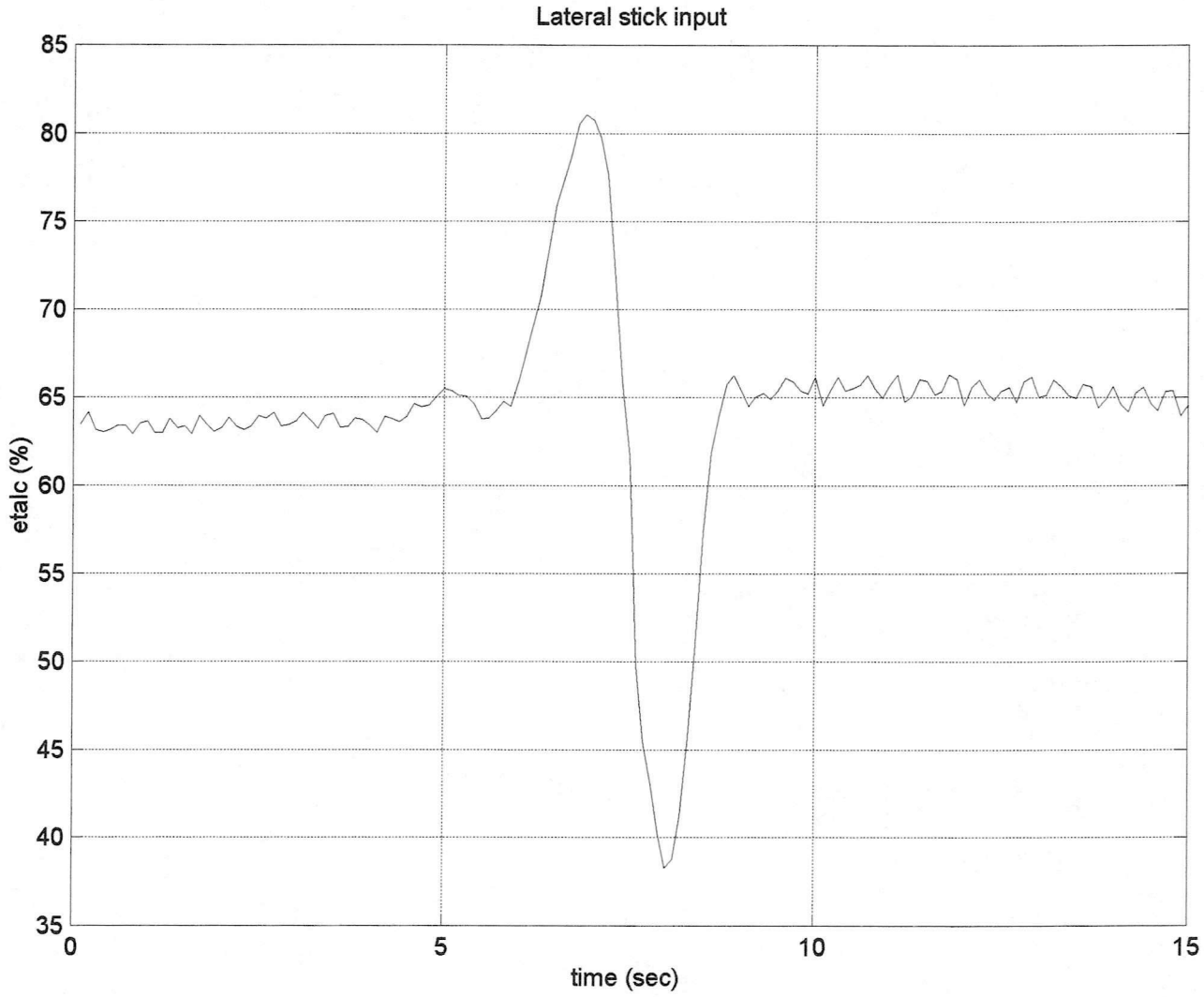

Figure 2 : Lateral stick input for case 2

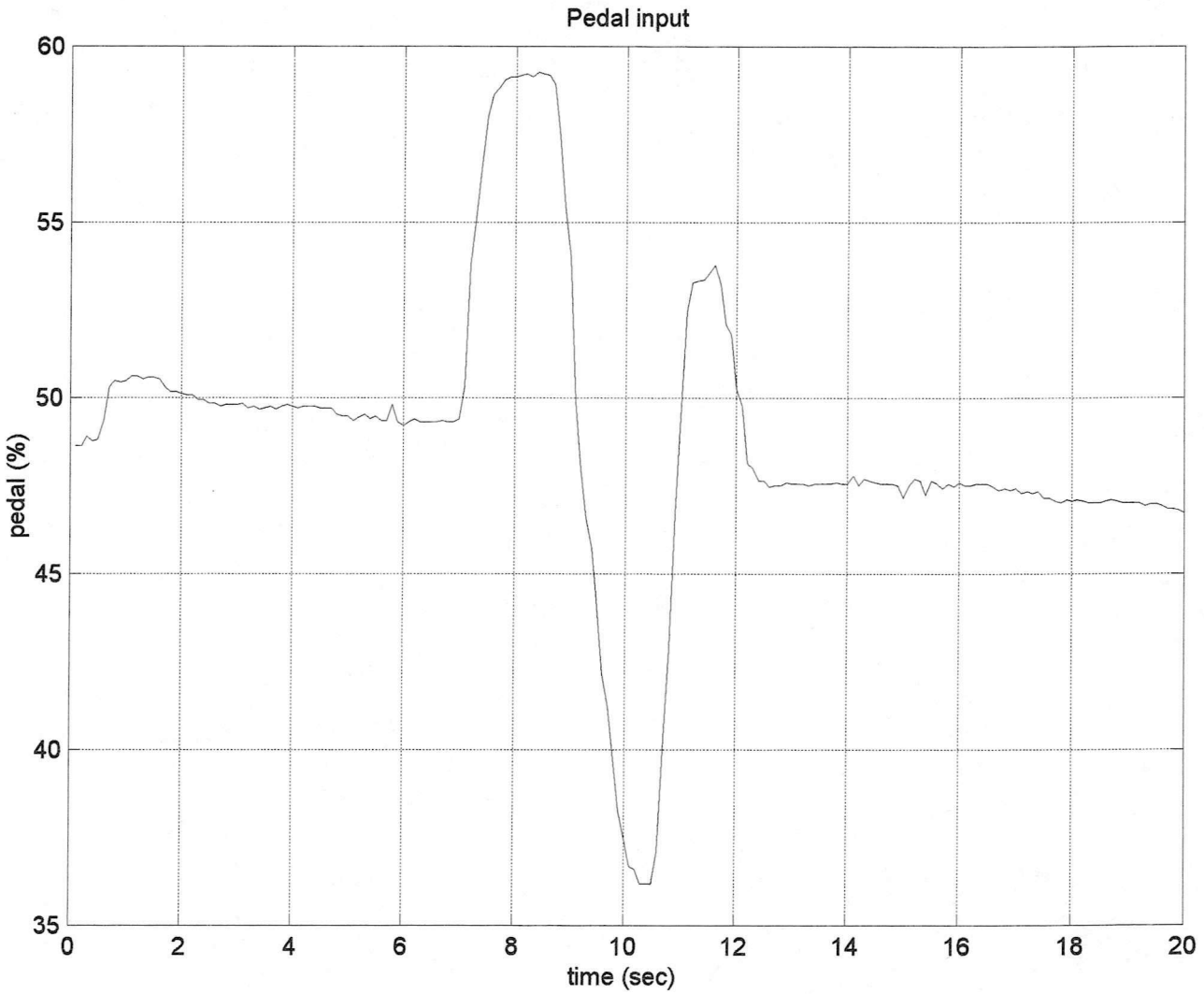

Figure 3 : Pedal input for case 3

Appendix A

Screen output for case 1

```
R^2 =
   0.5333

Xu =
   0.0980

error =
   0.0360

Xw =
  -0.2612

error =
   0.0745

Xq =
   0.3008

error =
   2.0755

Xtht =
  -9.0138

error =
   1.2358

Xhs =
   0.0167

error =
   0.0204

Xwm =
  -0.0407

error =
   0.0074

press any key..
R^2 =
   0.7087

Mu =
   0.0230

error =
   0.0017

Mw =
  -0.0465

error =
   0.0035
```

```
Mq =
  -1.1690

error =
   0.0971

Mtht =
  -0.2032

error =
   0.0578

Mhs =
   0.0258

error =
  9.5274e-004

Mwm =
  -0.0013

error =
  3.4777e-004

press any key..
R^2 =
   0.3509

Zu =
  -0.1307

error =
   0.0391

Zw =
  -0.4932

error =
   0.0810

Zq =
   27.0810

error =
   2.2571

Ztht =
   5.3825

error =
   1.3439

Zhs =
  -0.1045

error =
   0.0222

Zwm =
```

-0.0644

error =
0.0081

press any key..
R^2 =
0.8958

Omu =
1.4094

error =
0.0804

Omw =
5.7536

error =
0.1666

Omq =
4.0545

error =
4.6387

Omtht =
-5.4936

error =
2.7620

Omhs =
0.3098

error =
0.0455

Omwm =
-0.0692

error =
0.0166

Appendix B

Screen output for case 2

```
R^2 =
   0.9373

Yv =
  -0.0922

error =
   0.0176

Yr =
  -26.8604

error =
   0.5141

Yhtaped =
  -0.0446

error =
   0.0126

press any key..

R^2 =
   0.8959

Nv =
   0.0713

error =
   0.0013

Nr =
  -0.7032

error =
   0.0372

Nhtaped =
   0.0270

error =
  9.1325e-004
```

Appendix C

Screen output for case 3

```
R^2 =
   0.8778

Lv =
   0.0388

error =
   0.0047

Lp =
  -3.0562

error =
   0.0718

Lhtac =
   0.0732

error =
   0.0014
```

Appendix D

Matlab routine listing

```
function  [Asys,Bsys,Csys,Dsys]=solve(Amod,N,fr);

% Purpose : This routine is used in order to estimate the
% derivatives of a gyroplane using a least squares approach.
% Author : Vassilios M. Spathopoulos
% Date : 16/6/97



n=size(Amod);
w(1:N/2+1)=2*pi*(0:N/2)*10/N;
w(N/2+2:N)=2*pi*(5-(N/2+1:N-1)*10/N);
k=1;

% calculate frequency vector

while w(k)<2*pi*fr,
k=k+1;
end;
w=w(1:k);
w=w';

% define regression frequency range


Ac(:,1)=detrend(Amod(:,1),0);
Ac(:,2)=detrend(Amod(:,2),0);
Ac(:,3)=detrend(Amod(:,3),0);
Ac(:,4)=detrend(Amod(:,4),0);
Ac(:,5)=detrend(Amod(:,5),0);
Ac(:,6)=detrend(Amod(:,6),0);
Ac(:,7)=detrend(Amod(:,7),0);
Ac(:,8)=detrend(Amod(:,8),0);
Ac(:,9)=detrend(Amod(:,9),0);
Ac(:,10)=detrend(Amod(:,10),0);
Ac(:,11)=detrend(Amod(:,11),0);
Ac(:,12)=detrend(Amod(:,12),0);
Ac(:,13)=detrend(Amod(:,13),0);
Ac(:,14)=detrend(Amod(:,14),0);
Ac(:,15)=detrend(Amod(:,15),0);


% remove mean


Acf(:,1)=fft(Ac(:,1),N);
Acf(:,2)=fft(Ac(:,2),N);
Acf(:,3)=fft(Ac(:,3),N);
Acf(:,4)=fft(Ac(:,4),N);
Acf(:,5)=fft(Ac(:,5),N);
Acf(:,6)=fft(Ac(:,6),N);
Acf(:,7)=fft(Ac(:,7),N);
Acf(:,8)=fft(Ac(:,8),N);
Acf(:,9)=fft(Ac(:,9),N);
Acf(:,10)=fft(Ac(:,10),N);
```

```
Acf(:,11)=fft(Ac(:,11),N);
Acf(:,12)=fft(Ac(:,12),N);
Acf(:,13)=fft(Ac(:,13),N);
Acf(:,14)=fft(Ac(:,14),N);
Acf(:,15)=fft(Ac(:,15),N);

% calculate discrete Fourier transform


Ar(:,1)=real(Acf(1:k,1));
Ar(:,2)=real(Acf(1:k,2));
Ar(:,3)=real(Acf(1:k,3));
Ar(:,4)=real(Acf(1:k,4));
Ar(:,5)=real(Acf(1:k,5));
Ar(:,6)=real(Acf(1:k,6));
Ar(:,7)=real(Acf(1:k,7));
Ar(:,8)=real(Acf(1:k,8));
Ar(:,9)=real(Acf(1:k,9));
Ar(:,10)=real(Acf(1:k,10));
Ar(:,11)=real(Acf(1:k,11));
Ar(:,12)=real(Acf(1:k,12));
Ar(:,13)=real(Acf(1:k,13));
Ar(:,14)=real(Acf(1:k,14));
Ar(:,15)=real(Acf(1:k,15));


Ai(:,1)=imag(Acf(1:k,1));
Ai(:,2)=imag(Acf(1:k,2));
Ai(:,3)=imag(Acf(1:k,3));
Ai(:,4)=imag(Acf(1:k,4));
Ai(:,5)=imag(Acf(1:k,5));
Ai(:,6)=imag(Acf(1:k,6));
Ai(:,7)=imag(Acf(1:k,7));
Ai(:,8)=imag(Acf(1:k,8));
Ai(:,9)=imag(Acf(1:k,9));
Ai(:,10)=imag(Acf(1:k,10));
Ai(:,11)=imag(Acf(1:k,11));
Ai(:,12)=imag(Acf(1:k,12));
Ai(:,13)=imag(Acf(1:k,13));
Ai(:,14)=imag(Acf(1:k,14));
Ai(:,15)=imag(Acf(1:k,15));

% seperate to real and imaginary parts

clc;
choice=menu('What type of analysis ?','Longitudinal','Lateral');

if choice==1

% longitudinal analysis
A=[Ar(2:k,1)    Ar(2:k,3)    Ar(2:k,7)    Ar(2:k,5)         Ar(2:k,12)
Ar(2:k,15);Ai(2:k,1)  Ai(2:k,3)  Ai(2:k,7)  Ai(2:k,5)     Ai(2:k,12)
Ai(2:k,15)];
s=size(A);
B=[-w(2:k).*Ai(2:k,1);w(2:k).*Ar(2:k,1)];
V=eye(s(1))*1;
[X(1:6),dy]=lscov(A,B,V);
X=X';
disp('R^2 = ');
disp(((A*X)'*(A*X)/(B'*B)))
```

```
disp('Xu = ');
disp(X(1))

disp('error =');
disp(dy(1));

disp('Xw = ');
disp(X(2))

disp('error =');
disp(dy(2));


disp('Xq = ');
disp(X(3))

disp('error =');
disp(dy(3));


disp('Xtht = ');
disp(X(4))

disp('error =');
disp(dy(4));


disp('Xhs = ');
disp(X(5))

disp('error =');
disp(dy(5));


disp('Xwm = ');
disp(X(6))

disp('error =');
disp(dy(6));

disp('press any key..');
pause;
clc;


A=[Ar(2:k,1)    Ar(2:k,3)    Ar(2:k,7)    Ar(2:k,5)        Ar(2:k,12)
Ar(2:k,15);Ai(2:k,1)   Ai(2:k,3)   Ai(2:k,7)   Ai(2:k,5)     Ai(2:k,12)
Ai(2:k,15)];
B=[-w(2:k).*Ai(2:k,7);w(2:k).*Ar(2:k,7)];
V=eye(s(1));

[X(7:12),dy]=lscov(A,B,V);


disp('R^2 = ');
disp(((A*X(7:12))'*(A*X(7:12))/(B'*B)))

disp('Mu = ');
disp(X(7))

disp('error =');
```

```
disp(dy(1));


disp('Mw = ');
disp(X(8))

disp('error =');
disp(dy(2));


disp('Mq = ');
disp(X(9))

disp('error =');
disp(dy(3));


disp('Mtht = ');
disp(X(10))

disp('error =');
disp(dy(4));


disp('Mhs = ');
disp(X(11))

disp('error =');
disp(dy(5));


disp('Mwm = ');
disp(X(12))

disp('error =');
disp(dy(6));

disp('press any key..');
pause;
clc;

A=[Ar(2:k,1)    Ar(2:k,3)    Ar(2:k,7)    Ar(2:k,5)         Ar(2:k,12)
Ar(2:k,15);Ai(2:k,1)  Ai(2:k,3)  Ai(2:k,7)  Ai(2:k,5)     Ai(2:k,12)
Ai(2:k,15)];
B=[-w(2:k).*Ai(2:k,3);w(2:k).*Ar(2:k,3)];
V=eye(s(1));
[X(13:18),dy]=lscov(A,B,V);

disp('R^2 = ');
disp(((A*X(13:18))'*(A*X(13:18))/(B'*B)))

disp('Zu = ');
disp(X(13))

disp('error =');
disp(dy(1));


disp('Zw = ');
disp(X(14))
```

```
disp('error =');
disp(dy(2));


disp('Zq = ');
disp(X(15))

disp('error =');
disp(dy(3));


disp('Ztht = ');
disp(X(16))

disp('error =');
disp(dy(4));


disp('Zhs = ');
disp(X(17))

disp('error =');
disp(dy(5));


disp('Zwm = ');
disp(X(18))

disp('error =');
disp(dy(6));

disp('press any key..');
pause;
clc;


A=[Ar(2:k,1)    Ar(2:k,3)    Ar(2:k,7)    Ar(2:k,5)        Ar(2:k,12)
Ar(2:k,15);Ai(2:k,1)   Ai(2:k,3)   Ai(2:k,7)   Ai(2:k,5)     Ai(2:k,12)
Ai(2:k,15)];
B=[-w(2:k).*Ai(2:k,15);w(2:k).*Ar(2:k,15)];
V=eye(s(1));
[X(19:24),dy]=lscov(A,B,V);


disp('R^2 = ');
disp(((A*X(19:24))'*(A*X(19:24))/(B'*B)))

disp('Omu = ');
disp(X(19))

disp('error =');
disp(dy(1));


disp('Omw = ');
disp(X(20))

disp('error =');
disp(dy(2));
```

```
disp('Omq = ');
disp(X(21))

disp('error =');
disp(dy(3));


disp('Omtht = ');
disp(X(22))

disp('error =');
disp(dy(4));


disp('Omhs = ');
disp(X(23))

disp('error =');
disp(dy(5));


disp('Omwm = ');
disp(X(24))

disp('error =');
disp(dy(6));


% use least squares approach to estimate and display longitudinal
derivatives
% and associated errors

Asys=[X(1) X(2) X(3) X(4) X(6);X(13) X(14) X(15) X(16) X(18);X(7)
X(8) X(9) X(10) X(12);0 0 1 0 0; X(19) X(20) X(21) X(22) X(24)];
Bsys=[X(5) X(17) X(11) 0 X(23)]';
Csys=zeros(5);
Csys(1,1)=1;
Dsys=zeros(5,1);


% define state space matrice

else

% lateral analysis
A=[Ar(2:k,2) Ar(2:k,8) Ar(2:k,14);Ai(2:k,2) Ai(2:k,8) Ai(2:k,14)];
s=size(A);
B=[-w(2:k).*Ai(2:k,2);w(2:k).*Ar(2:k,2)];
V=eye(s(1));

[X(1:3),dy]=lscov(A,B,V);
X=X';
disp('R^2 = ');
disp(((A*X)'*(A*X)/(B'*B)))

disp('Yv = ');
disp(X(1))

disp('error =');
disp(dy(1));
```

```
disp('Yr = ');
disp(X(2))

disp('error =');
disp(dy(2));


disp('Yhtaped = ');
disp(X(3))

disp('error =');
disp(dy(3));

disp('press any key..');
pause;
clc;


A=[Ar(2:k,2) Ar(2:k,6) Ar(2:k,13);Ai(2:k,2) Ai(2:k,6) Ai(2:k,13)];
B=[-w(2:k).*Ai(2:k,6);w(2:k).*Ar(2:k,6)];
V=eye(s(1));

[X(4:6),dy]=lscov(A,B,V);

disp('R^2 = ');
disp(((A*X(4:6))'*(A*X(4:6))/(B'*B)))

disp('Lv = ');
disp(X(4))

disp('error =');
disp(dy(1));

disp('Lp = ');
disp(X(5))

disp('error =');
disp(dy(2));

disp('Lhtac = ');
disp(X(6))

disp('error =');
disp(dy(3));

disp('press any key..');
pause;
clc;


A=[Ar(2:k,2) Ar(2:k,8) Ar(2:k,14);Ai(2:k,2) Ai(2:k,8) Ai(2:k,14)];
B=[-w(2:k).*Ai(2:k,8);w(2:k).*Ar(2:k,8)];
V=eye(s(1));

[X(7:9),dy]=lscov(A,B,V);

disp('R^2 = ');
disp(((A*X(7:9))'*(A*X(7:9))/(B'*B)))

disp('Nv = ');
disp(X(7))
```

```
disp('error =');
disp(dy(1));

disp('Nr = ');
disp(X(8))

disp('error =');
disp(dy(2));

disp('Nhtaped = ');
disp(X(9))

disp('error =');
disp(dy(3));


end;

% use least squares approach to estimate and display lateral
derivatives
% and associated errors

return
```